Fuzzing Large Language Models[1]

Esben Kran
Apart Research

Abstract

Language models have shown significant non-human and surprising behavior given both specific adversarial and stochastic prompts. In this work, we use a 14B parameter RNN to adversarially generate 'fuzzing' prompts, prompts intentionally designed to elicit surprising and off-distribution responses. We then use the same model to evaluate the surprisal of the output given the prompt on a scale of 1 to 5. Our work presents the next steps towards elicitation and identification of out-of-distribution failure cases for LLMs, following valuable work from the AI safety research community. These are shared as a dataset at https://github.com/esbenkc/verification-jam.

Keywords: Scalable oversight, benchmarks, ML safety

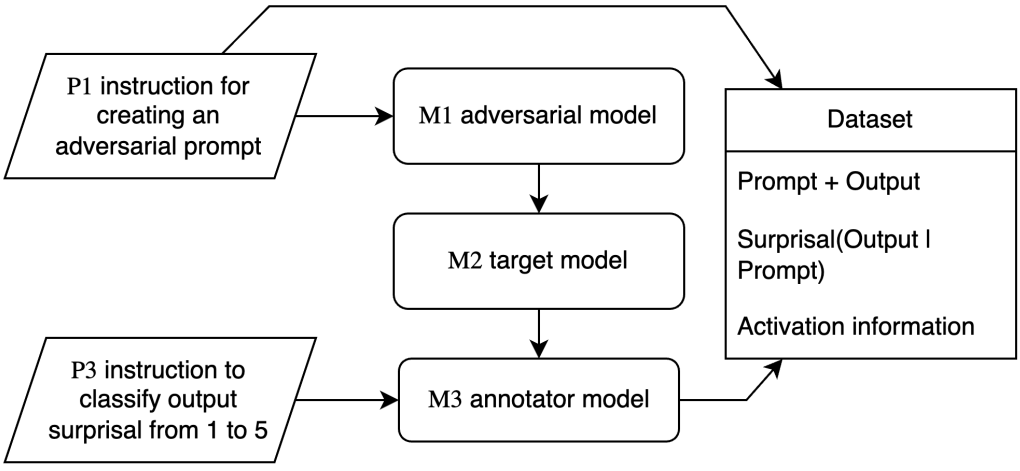

Figure 1 – Diagram of the prompt and model pathway of our fuzzing dataset

[1] Research conducted at the Apart Research Alignment Jam #8 (Safety Verification), 2023 (see https://alignmentjam.com/jam/verification)

1. Introduction

Fuzzing is a critical technique in software testing that involves the execution of a program using generated inputs that may be syntactically or semantically malformed. These inputs, derived from an input space that extends beyond the expected program parameters, serve to expose latent vulnerabilities or aberrations in the system.

There are two primary methods of fuzzing: white-box and black-box. White-box fuzzing, introduced by Godefroid in 2007, uses the internals of the program to generate fuzzing examples. While this method can be slow, it provides insightful perspectives, particularly in the context of neural networks. On the other hand, black-box fuzzing, also known as IO-driven or data-driven testing, does not rely on the internal structure of the program. This technique becomes especially intriguing when applied to large language models (LLMs), with LLMs themselves being employed to generate the testing examples.

In the field of AI safety, detecting and mitigating the unexpected behaviors of language models are of paramount importance. This is not merely due to the potential exploit vulnerabilities, but also to avoid general misbehaviors during the deployment phase. As Rumbelow & Watkins (2023) have demonstrated, the structure of the training data, particularly in the long tail of token probabilities, can lead to a network semantically misinterpreting tokens. This exemplifies how the inherent security risks in language models can manifest, reinforcing the need for effective fuzzing strategies.

2. Methods

In this study, we employ the newly developed RWKV architecture by EleutherAI (2023), incorporating approximately 540M parameters - a notable reduction compared to the largest model, which employs 14B parameters. This choice offers an effective balance between computational efficiency and model complexity.

The experimental setup involved the use of multiple prompt histories. Specifically, we constructed three histories, H1, H2, and H3. H1 comprises the P1 input and any output from model M1 to P1, with M1_out denoting the isolated output from M1. H2 was composed of M1_out and M2's output to P1_out, defined as M2_out. Lastly, H3 encapsulated P3, which was formatted with M1_out and M2_out, and included M3's output given this formatted P3. For H3, we restricted the output to a single digit numerical value, enforcing a specific constraint on the model.

This approach paves the way towards the realization of automated fuzzing, offering a structured and efficient method for testing large language models. However, it's worth noting that this research remains in its early stages. A complete run of the model was not achieved due to unforeseen circumstances - a power outage during testing at the airport. Future work will aim to rectify this situation and provide comprehensive results on the effectiveness of this method.

3. Results & discussion

Upon analysis of the results, we found that the 540M parameter RWKV was not capable of executing or evaluating fuzzing effectively. This underscores the challenge of task complexity in relation to model size, indicating that further model optimization or an increase in parameters might be required for successful execution.

Furthermore, our observations highlight the crucial role of the P1 prompt in the process. Improvements in this area could be achieved through a chatbot Reinforcement Learning from Human Feedback (RLHF) step, which could provide the model with a more dynamic and interactive learning environment.

However, it is worth noting that our 100-line implementation of the RWKV may have been overly simplistic for this specific scenario. Future implementations might need to incorporate more complexity to handle the diverse challenges that arise in fuzzing.

Moving forward, we have identified several next steps to continue this research. A manual approach can be tested, where three separate ChatGPT windows are used to simulate the process. Additionally, using the GPT-4 API could provide another avenue to evaluate the effectiveness of the model's outputs. Furthermore, to gain more insights into the model's behavior during fuzzing attempts, we could save the network's activations for subsequent analysis.

For additional details on the future direction of this research and other planned improvements, we invite readers to visit our repository at https://github.com/esbenkc/verification-jam. There, we have outlined further steps to continue pushing the boundaries of this novel approach to language model fuzzing.

4. Discussion and Conclusion

In conclusion, fuzzing emerges as a crucial method in AI safety, providing insights into unexpected behaviors of language models and helping identify potential vulnerabilities. Its application to large language models, however, is in its infancy and is laden with challenges, as seen with our experience with the 540M parameter RWKV model. Our exploration underscored the complexity of the task, the importance of prompt quality, and potential limitations of simplistic implementations. As AI technologies evolve, refining and optimizing fuzzing techniques will be paramount to ensure robust and secure language models. Nonetheless, the journey towards efficient fuzzing is as challenging as it is promising, meriting rigorous research and innovative approaches.